

基于 PLI 算法的 AAC 反量化和 PNS 解码模块设计

张红升^{1,2}, 王国裕^{1,2}, 陆明莹¹, 蒋 涛¹, 郭光宇¹

(1 重庆邮电大学 微电子工程重点实验室, 重庆 400065; 2 西安交通大学 微电子学系, 陕西 西安 710049)

摘 要: 针对 AAC(Advanced Audio Coding)解码器中具有复杂运算因而难以硬件实现的反量化模块和知觉噪音替代(Perceptual Noise Substitution, PNS)模块, 提出用分段线性插值(Partition and Linear Interpolation, PLI)进行近似计算的方法. 其基本原理是将复杂的曲线分割成多个区间, 在每个区间内用线性插值的方法进行近似. 利用该方法设计的反量化模块和 PNS 解码模块获得了较高的运算精度, 最大运算误差率分别为 $2e-4$ 和 $5e-3$, 全部运算可分解为简单的查表、乘法、加法和移位运算, 易于硬件实现. FPGA 验证结果表明, 反量化模块仅耗用了 45 个逻辑单元和 2 304 ROM bits, PNS 解码模块仅耗用了 21 个逻辑单元和 144 ROM bits, 且解码音质良好, 没有产生可感知的噪音.

关键词: AAC; 分段线性插值; 反量化; 知觉噪音替代

中图分类号: TN492

文献标识码: A

文章编号: 1000-7180(2012)08-0001-05

Design of AAC Inverse Quantization and PNS Decoder Based on PLI method

ZHANG Hong-sheng^{1,2}, WANG Guo-yu^{1,2}, LU Ming-ying¹, JIANG Tao¹, GUO Guang-yu¹

(1 Key Lab of Microelectronic Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China; 2 Department of Microelectronics, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: A partition and linear interpolation (PLI) method is proposed, in order to simplify the complicated algorithm of the inverse quantization and perceptual noise substitution (PNS) blocks of the advanced audio coding (AAC) decoder. The principle of the PLI method is to divide a complicated function into several regions and apply linear interpolation in each region to obtain the approximate value of the function. The simplified algorithms achieved low max error rates as $2e-4$ and $5e-3$ for inverse quantization and PNS decoder respectively. The computations comprise only look-up table (LUT), multiplication, add, and shift operations, thus can be easily implemented in hardware. The FPGA verification results showed the inverse quantization block consumes only 45 logic elements and 2 304 ROM bits, and the PNS decoder consumes only 21 logic elements and 144 ROM bits. The audio quality is good and no perceptible noise is generated.

Key words: AAC; partition and linear interpolation; inverse quantization; PNS

1 引言

AAC(Advanced Audio Coding)是新一代音频压缩标准,可以比 MP3 文件缩小 30%,同时支持多达 48 个音轨,有更多种采样率/比特率组合以及更高的解码效率^[1]. 由于 AAC 编码的灵活性和高效

性,许多现代通信系统都采用 AAC 作为音频压缩标准,如 DAB(Digital Audio Broadcasting),DRM(Digital Radio Mondiale)等. 这些系统的许多接收终端是便携式的,因而对功耗提出了严格要求.

目前大多数 AAC 解码芯片都采用内嵌 CPU 并运行软件的方式解码^[2-4]. 这种方式可以方便地对

系统进行升级和扩展,但由于 AAC 解码算法的复杂性,CPU 通常需要运行在较高的主频(最低 35 MHz,文献[2-3]),因而功耗难以降低。文献[5-7]采用了 ASIC 方式设计了硬件 AAC 解码器的全部或部分电路,工作主频分别降低到 5 MHz 和 1.3 MHz。其中文献[6]将硬件 AAC 解码器与软件 AAC 解码器做了详细的对比,从中可以看出硬件解码器在功耗上有明显的优势。本课题组在前期研究^[8]中,采用 ASIC 设计方式,完成了针对便携式 DAB 接收机的硬件 AAC 解码器,并通过了 FPGA 验证。

本文在硬件 AAC 解码器设计中采用分段线性插值法(Partition and Linear Interpolation, PLI),并应用于反量化和知觉噪音替代(Perceptual Noise Substitution, PNS)模块,以获得较高的运算精度。

2 算法原理

PLI 算法的基本原理是将 $f(x)$ 的自变量 x 分成 N 个区间,区间的端点分别为 $x_0, x_1, x_2 \cdots x_n$,那么对任意的 $x \in [x_{n-1}, x_n]$,可以通过线性插值的方法求得 $f(x)$ 的近似值,如图 1 所示。

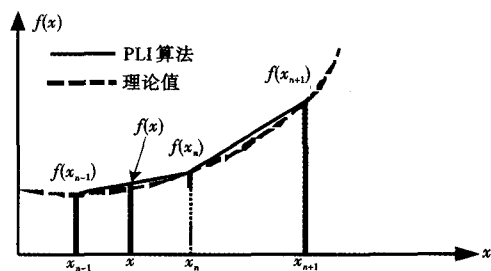


图 1 PLI 算法原理图

PLI 算法的基本公式为

$$f(x) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}(x - x_{n-1}) + f(x_{n-1}) \quad (1)$$

通过适当选取 x_n 的取值,并把对应的 $f(x_n)$ 预先存储起来,再按照式(1)进行查表和乘加运算,即可得到任意 x 所对应的 $f(x)$ 值。

PLI 算法的精度一方面取决于 n 的数值, n 越大则精度越高。另一方面也取决于区间 $[x_{n-1}, x_n]$ 的大小,在 $f(x)$ 的导数 $f'(x)$ 较大的区域,应划分较小的区间;在 $f'(x)$ 较小的区域,则可以划分较大的区间。适当地选择区间的大小和数目,可以在计算精度和硬件复杂度之间获得较好的折中。

3 AAC 反量化模块的设计

反量化运算是 AAC 解码器的主要模块,目的是将比特流解析得到的频域子带样点重新还原为量化之前的数值,其运算量占全部 AAC 解码运算的 5%~10%。AAC 采用了非线性量化的方法,其反量化公式为

$$x_{dep} = \text{sign}(x_{quant}) \cdot |x_{quant}|^{4/3} \quad (2)$$

式中, x_{quant} 为反量化前的声音频谱值; $\text{sign}(x_{quant})$ 为谱线的符号,

$$\text{sign}(x_{quant}) = \begin{cases} 1, & x_{quant} \geq 0 \\ -1 & x_{quant} < 0 \end{cases} \quad (3)$$

从式(2)可以看出,反量化的主要运算是对 $|x_{quant}|^{4/3}$ 的计算。按照 AAC 标准, $|x_{quant}|$ 为 0~8191 的整数。式(2)中由于有指数运算,在软件 AAC 解码器中,一般是将 8192 个 $|x_{quant}|^{4/3}$ 值全部计算出来并存储在 ROM 中,通过查表的方法直接得到结果。但这么大的 ROM 在硬件设计中是难以接受的,因为这将占用较大的芯片面积。因此本文采用 PLI 算法来近似计算 x_{deq} 的值。

为较好地选取 $|x_{quant}|$ 的取值区间,统计了超过 200 套不同码率不同采样率的 AAC 节目,并对这些节目中 $|x_{quant}|$ 的取值进行了统计,统计结果如表 1 所示。从表 1 中可以看出,尽管 $|x_{quant}|$ 的取值范围很大,但实际上绝大部分 $|x_{quant}|$ 位于 $[0, 127]$ 之间,而且超过 128 的概率非常小。

表 1 $|x_{quant}|$ 的概率分布

$ x_{quant} $	概率/%	$ x_{quant} $	概率/%
0	85.143	6	0.076
1	12.340	7	0.047
2	1.428	8~15	0.106
3	0.478	16~127	0.021
4	0.228	128~511	0.000 013
5	0.131	>512	0

根据表 1 的统计结果,为保证音质,可以将 $[0, 127]$ 之间的 $|x_{quant}|$ 所对应的 $|x_{quant}|^{4/3}$ 值存储起来,用查表的方法直接求得。而对于 $[128, 8191]$ 之间的 $|x_{quant}|$,则对其先提取一个系数后,映射到 $[0, 127]$ 区间中,然后再通过 PLI 算法得到近似的 $|x_{quant}|^{4/3}$ 值。具体过程如下:

(1) 对于 $0 \leq |x_{quant}| < 128$,直接通过查表求出 $|x_{quant}|^{4/3}$;

(2) 对于 $128 \leq |x_{quant}| < 1024$,将 $|x_{quant}|$

除以8映射到 $[0, 127]$,并按如下规则进行运算:

$$|x_{\text{quant}}|^{4/3} \approx 16 \times \left\lfloor \frac{|x_{\text{quant}}|}{8} \right\rfloor^{4/3} + 2 \times \text{rem}\left(\frac{|x_{\text{quant}}|}{8}\right) \times \left(\left\lfloor \frac{|x_{\text{quant}}|}{8} \right\rfloor^{4/3} - \left\lfloor \frac{|x_{\text{quant}}|}{8} \right\rfloor^{4/3} \right) \quad (4)$$

式中, $\lceil x \rceil$ 运算符代表大于或等于 x 的最小整数, $\lfloor x \rfloor$ 代表小于或等于 x 的最大整数, $\text{rem}(x/y)$ 代表 x 除以 y 的余数.

(3) 对于 $1\,024 \leq |x_{\text{quant}}| < 8\,192$,将 $|x_{\text{quant}}|$ 除以64映射到 $[0, 127]$,并按如下规则进行运算:

$$|x_{\text{quant}}|^{4/3} \approx 256 \times \left\lfloor \frac{|x_{\text{quant}}|}{64} \right\rfloor^{4/3} + 4 \times \text{rem}\left(\frac{|x_{\text{quant}}|}{64}\right) \times \left(\left\lfloor \frac{|x_{\text{quant}}|}{64} \right\rfloor^{4/3} - \left\lfloor \frac{|x_{\text{quant}}|}{64} \right\rfloor^{4/3} \right) \quad (5)$$

通过以上处理, $|x_{\text{quant}}|$ 的取值首先被分成3个大的区域: $[0, 127]$, $[128, 1\,023]$, $[1\,024, 8\,191]$. 由于大部分 $|x_{\text{quant}}|$ 位于第一个区域,因此对这段区域采用查表方法实现,以保证最高的运算精度和最好的音质;对第二段区域,按间隔等于8来进行线性插值;对第三段区域,按间隔等于64来进行线性插值.这样概率越小的区域,间隔越大,从而在硬件复杂度和运算可靠性之间取得了较好的折中.

假设 $|x_{\text{quant}}|$ 在硬件上用 N 个比特来存储,即 $|x_{\text{quant}}| = \{B_N, B_{N-1}, \dots, B_1, B_0\}$,那么式(4)和式(5)中的各部分在硬件电路中分别对应:

$$\begin{cases} \left\lfloor \frac{|x_{\text{quant}}|}{8} \right\rfloor = \{B_N, B_{N-1}, \dots, B_3\} \\ \left\lfloor \frac{|x_{\text{quant}}|}{8} \right\rfloor = \{B_N, B_{N-1}, \dots, B_3\} + 1 \\ \left\lfloor \frac{|x_{\text{quant}}|}{64} \right\rfloor = \{B_N, B_{N-1}, \dots, B_6\} \\ \left\lfloor \frac{|x_{\text{quant}}|}{64} \right\rfloor = \{B_N, B_{N-1}, \dots, B_6\} + 1 \\ \text{rem}\left(\frac{|x_{\text{quant}}|}{64}\right) = \{B_2, B_1, B_0\} \\ \text{rem}\left(\frac{|x_{\text{quant}}|}{64}\right) = \{B_5, B_4, \dots, B_0\} \end{cases} \quad (6)$$

式(4)和式(5)中乘以2、4、16和64的运算在硬件上可以通过简单的移位操作来实现.因此式(4)和式(5)只需要2次查表、2次移位、1次减法和1次乘法运算即可实现.

4 PNS 解码模块的设计

PNS是一种以参数编码的方式模拟噪声的技术.当编码端检测出声音中有噪音分量时,只传输噪音的整体能量,而不传输噪音的具体波形.解码端用随机序列来模拟噪声,并调整随机序列的幅度使得噪音的总能量满足要求.PNS解码的关键是对随机序列调整系数 $scale$ 的计算:

$$scale = 2^{0.25 * \text{noise_nrg}} / \sqrt{\text{nrg}} \quad (7)$$

式中, noise_nrg 是应该添加的噪音能量值,由比特流解析得到; nrg 为当前添加的噪音的平均能量,也就是解码端产生的随机序列幅度平方的平均值.

式(7)中有开平方和求导运算,而且 nrg 取值的可能性很多,难以直接用查表法实现.为了在硬件上实现上述运算,将 nrg 表示成如下形式:

$$\text{nrg} = a \cdot 2^b \quad (8)$$

式中, $a \in [0.5, 1)$, b 为整数.假设 nrg 在硬件存储中其前导零的个数为 m ,那么就有 $b = m - 1$, $a = \text{nrg} \ll b$.

将式(8)代入到式(7)中得到:

$$scale = a^{-0.5} \cdot 2^{(\text{noise_nrg} - 2b)/4} \quad (9)$$

由于 noise_nrg 和 b 都是整数,所以 $(\text{noise_nrg} - 2b)/4$ 总可以表示成 $Q + R$,其中 Q 是正整数, R 是 $\{-1, -0.75, -0.5, -0.25\}$ 中的一个值.因此,式(9)还可以写成:

$$scale = (a^{-0.5}/4) \cdot 2^R \cdot 2^{Q+2} \quad (10)$$

式中, $a^{-0.5}/4$ 和 2^R 都 $\in [0.5, 1)$.由于 2^R 只有4个可能的值,可以很方便地用查表或多路选择器来实现,因此式(10)的重点是对 $a^{-0.5}/4$ 的处理.

令 $f(a) = a^{-0.5}/4$,那么可以采用如下的PLI算法对 $f(a)$ 进行处理.

(1) 对 $0.5 \leq a < 0.625$:

$$f(a) \approx f(0.5) + (a - 0.5) \times 8 \times [f(0.625) - f(0.5)] \approx 0.502\,9 - 0.298\,6 \times a \quad (11)$$

(2) 对 $0.625 \leq a < 0.75$:

$$f(a) \approx f(0.625) + (a - 0.625) \times 8 \times [f(0.75) - f(0.625)] = 0.454\,0 - 0.220\,4 \times a \quad (12)$$

(3) 对 $0.75 \leq a < 0.875$:

$$f(a) \approx f(0.75) + (a - 0.75) \times 8 \times [f(0.875) - f(0.75)] = 0.4172 - 0.1713 \times a \quad (13)$$

(4) 对 $0.875 \leq a < 1$:

$$f(a) \approx f(0.875) + (a - 0.875) \times 8 \times [f(1) - f(0.875)] = 0.3881 - 0.1381 \times a \quad (14)$$

上述 4 个公式中共有 8 个常数需要存储, 每一个公式的计算需要 2 次查表, 1 次乘法和 1 次加法. 再考虑乘以 2^R 和 2^{Q+2} 的操作, 那么 $scale$ 的计算还需要 1 次查表, 1 次乘法和 1 次移位操作. 这些操作在硬件上都比较容易实现, 而且一共只需要存储 12 个常数, 因此硬件开销很小.

5 测试

图 2 中给出了反量化模块中 $|x_{quant}|^{4/3}$ 的理论值与 PLI 算法得到的近似值的对比. 从图中可以看出, 两个算法的结果非常接近, 曲线基本上重合在一起. 图 3 中给出了 PLI 算法的误差率, 其计算公式为: 误差率 = (近似值 - 理论值) / 理论值. 从中可以看出, 最大误差率约为 $2e-4$, 略高于文献[6] ($1e-4$), 但本文只用了 128 个存储地址, 因而电路面积更小. 测试结果还表明, 本文与文献[6]的解码结果在音质上没有可感知的差别. 与文献[8]相比, 多了 1 次查表和乘法操作, 但获得了更高的运算精度. 由于乘法器在硬件设计中是多个运算模块共享的, 因此多出来的查表和乘法操作, 并不会显著增加电路的面积.

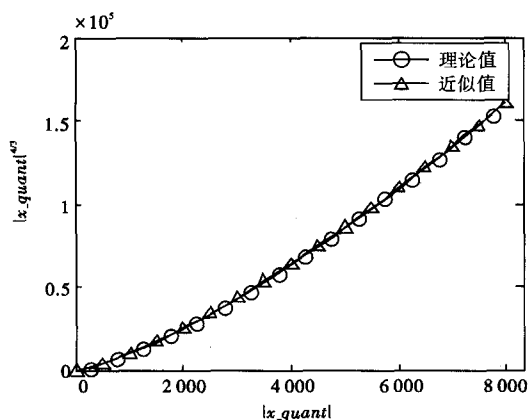


图 2 $|x_{quant}|^{4/3}$ 的理论值与近似值的对比

图 4 中给出了 PNS 解码模块中 $a^{-0.5/4}$ 的理论值与 PLI 算法得到的近似值的对比. 图 5 给出了 PLI 算法的误差率, 其最大值约为 $5e-3$. 由于 PNS

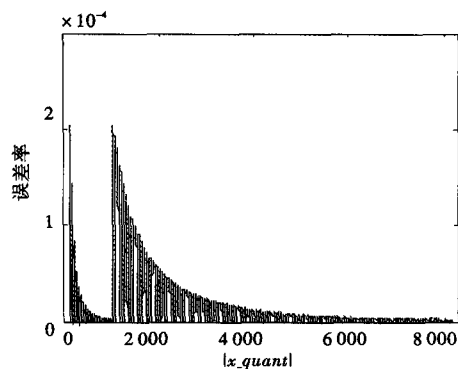


图 3 PLI 算法计算 $|x_{quant}|^{4/3}$ 的误差率

是噪音替代技术, 而人耳对噪音的敏感程度较低, 因此这个精度完全满足设计要求.

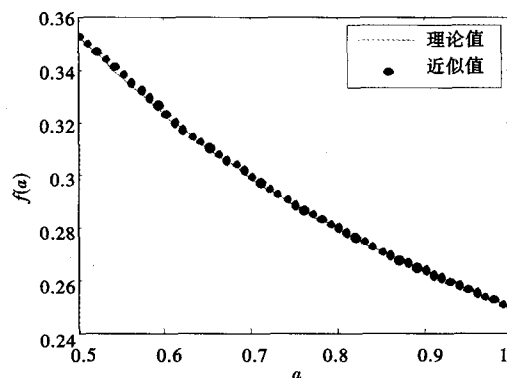


图 4 $a^{-0.5/4}$ 的理论值与近似值的对比

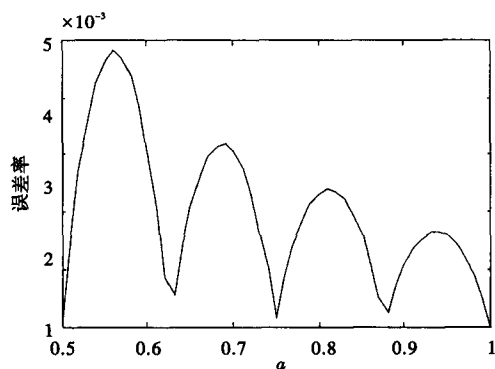


图 5 PLI 算法计算 $a^{-0.5/4}$ 的误差率

上述算法在 Altera EP3C120 FPGA 开发板上进行了验证. 反量化模块仅耗用 45 个逻辑单元 (不包括加法和乘法器电路, 因为这些电路是和其他模块共享, 下同) 和 2 304 ROM bits. PNS 解码模块仅耗用 21 个逻辑单元和 144 ROM bits. 通过对大量 AAC 节目的测试结果表明, 上述算法获得了良好的音质, 与软件 AAC 解码器相比, 没有产生可感知的噪音.

6 结束语

文中从ASIC设计角度出发,利用PLI算法对AAC解码器中的反量化模块和PNS解码模块进行了简化,在保证运算精度的前提下,将复杂的运算分解为简单的查表、加法、乘法和移位操作的组合,易于硬件实现。最后通过仿真和FPGA验证,证明了简化后的算法有较高的运算精度,而且电路面积小,为硬件AAC解码器的设计提供了重要参考。

参考文献:

- [1] Herre J, Dietz M. MPEG-4 high-efficiency AAC coding [J]. IEEE Signal Processing Magazine, 2008, 25 (3): 137-142.
- [2] Keun-Sup Lee, Young Cheol Park, Dae Hee Youn. Software optimization of the MPEG audio decoder using a 32-bit MCU RISC processor [J]. IEEE Transactions on Consumer Electronics, 2002, 48(3): 671-676.
- [3] 陈东晓,周建,张华,等. 一种媒体增强的可配置 RISC 微处理器核设计[J]. 微电子学与计算机, 2006, 23 (10): 197-200.
- [4] Yan Ming, Hu Guorong, Wang Jia. Design and implementation of MPEG-4 AAC decoder on ARM Embedded system for CMMB receiver [C] // International Conference on Management and Service Science. Wu-

han, China: IEEE Wuhan Society, 2009:1-3.

- [5] Peilin Liu, Lingzhi, Liuning Deng, et al. VLSI implementation for portable application oriented MPEG-4 audio codec [C] // IEEE International Symposium on Circuits and Systems. New Orleans, USA: IEEE, 2007:777-780.
- [6] Tsung-Han Tsai, Chun-Nan Liu. Low-power system design for MPEG-2/4 AAC audio decoder using pure ASIC approach [J]. IEEE Transactions on Circuits and Systems—I: Regular Papers, 2009, 56(1): 144-155.
- [7] 赵喜林,林涛. AAC 音频解码器中 TNS 模块的设计及其 ASIC 实现[J]. 微电子学与计算机, 2006, 23 (1): 174-180.
- [8] 陆明莹,张丽丽,王国裕,等. 低功耗便携式数字音频广播接收机中 AAC LC 解码器的设计优化[J]. 电子与信息学报, 2011, 33 (5): 1229-1232.

作者简介:

张红升 男,(1980—),博士研究生,讲师.研究方向为DAB/DMB解码芯片和整机方案的设计.

王国裕 男,(1954—),教授,博士生导师.研究方向为音频压缩、DAB/DMB芯片和系统设计.

陆明莹 女,(1954—),教授.研究方向为数字信号处理、DAB/DMB芯片和系统设计.